

Real-Time Programming

Summer Semester 08

Assignment #2

Homepage: http://www.iste.uni-stuttgart.de/ps/Lehre/SS2008/V_RTP/

Discussion of Solutions: May 8 at 9:45 in V38.02

1 Reference Counting I

Given is the following program in Ada95. Parameter passing is done by value. The compiler implements “Reference Counting” for objects on the heap (transparently for the user).

```
( 0) type Node;
( 1) type Node_Acc is access Node;
( 2) type Node is
( 3)     record
( 4)         Content : Integer;
( 5)         Next    : Node_Acc;
( 6)     end record;
( 7) function test return Node_Acc
( 8)     P,Q,R : Node_Acc;
( 9)     procedure Inner ( A, B, C : Node_Acc ) is
(10)     begin
(11)         declare
(12)             S : Node_Acc := new Node'(8, C); -- Objekt 04
(13)         begin
(14)             P := B;
(15)             S := A;
(16)             B.Next := C;
(17)             ... -- Accesses, but no modifications of pointers
(18)         end;
(19)         return B.Next.Next;
(20)     end Inner;
(21) begin
(22)     P := new Node'(5, null); -- Object 01
(23)     Q := new Node'(6, null); -- Object 02
(24)     Q.Next := P;
(25)     R := new Node'(7, Q);    -- Object 03
(26)     R := Inner(P, R, Q);
(27)     return;
(28) end test;
(29) test;
```

- a) Insert the values of the reference counters after the execution of the corresponding line into the table. All effects of actions invoked by “Reference Counting” have to be considered.

Hint: Draw figures that show the state of the objects and the pointers at specific times during the execution.

	O1	O2	O3	O4
After line (16)				
After line (18)				
After line (26)				

- b) Insert the number of the line where the deallocation due to “Reference Counting” occurs into the table. Mark the entry with 'X' if the object is not released from memory

	O1	O2	O3	O4
Deallocation after line				

2 Reference Counting II

You are the heap expert in a company that uses Ada95 for real-time programming. One specific program runs out of memory due to the heap garbage problem (non-released memory blocks to which no pointer is referring to anymore). You have traced the problem to a recursive record data structure called `Rec`.

Because there is no automatic support for garbage collection available, you decide to overcome the problem by manually applying the *reference counting* approach for this data structure. Then, all programmers have to add code according to your guidelines in order to make the reference counting strategy work.

1. The Ada package specification at

```
http://www.iste.uni-stuttgart.de/ps/Lehre/SS2008/V_RTP/a2/rc.
ads
```

declares the recursive data structure `Rec` along with the reference counting interface.

Implement the interface in the corresponding package body.

Hint: If you run into trouble, a partial solution is already provided in the file `rc.adb`.

2. Specify the programmer guidelines by stating all relevant programming language constructs along with the necessary calls to the reference counting interface.

Hint: Don't forget parameter passing and implicit deallocations.

3. The Ada code given at

```
http://www.iste.uni-stuttgart.de/ps/Lehre/SS2008/V_RTP/a2/rctest.
adb
```

uses the `Rec` data type, but so far lacks the calls to the reference counting interface. Please insert the appropriate calls, so that garbage is detected and dangling references are avoided.

Hint: You should get a count of 2001 deallocated objects.

4. Which advices do you give to programmers in order to inform them about limitations of the reference counting scheme and what do you recommend to overcome these limitations?

3 Exception Handling and Interfaces

You have just been hired by a new start-up company called `Stacks-R-Us.com`. The company has developed a stack implementation and a driver program (for testing) that allows the user to manipulate the stack interactively.

The stack is implemented as a package in Ada95 in the files `stacks.ads` (package specification) and `stacks.adb` (package body). The driver is implemented in `stackmain.adb`.

The source can be obtained from the following URL:

http://www.iste.uni-stuttgart.de/ps/Lehre/SS2008/V_RTP/a2

1. The current implementation has not been documented by your predecessor. Your first task is to add meaningful comments to the `stacks.ads` file.

Hint: Reread section 1.3 of the lecture notes.

2. The current implementation has two major deficiencies:
 - Misusages of the stack are not checked and no user-defined exceptions are raised when a stack is incorrectly used.
 - The driver program that uses the stack package is not prepared to handle exceptions in package stacks.

Since the company wants to sell the stack implementation as a COTS, you have to fix these deficiencies by adding appropriate exceptions (in the package specification) and code that checks for misusages and raises the corresponding exceptions if necessary (in the package body). Moreover, extend the driver program to cope with exceptions raised by the stacks package. Altogether, your program must not fail due to wrong user input¹.

4 Exception Handling Control Flow

Given the following Ada95 fragment, which output is generated if procedure `EH` is called?

¹You can assume that the user always inputs a number and does not input arbitrary strings.

```

with Ada.Text_IO;
use  Ada.Text_IO;

procedure EH is
  e1, e2 : exception;

  procedure f is
  begin
    Put ("L");
    raise e1;
    Put ("O");
  exception
    when e1 => Put ("90"); raise e2; Put ("09");
    when e2 => Put ("50"); raise e1; Put ("60");
    when others => Put ("12"); raise; Put ("34");
  end f;

  procedure g is
  begin
    Put ("A");
    f;
    Put ("B");
  exception
    when others => Put ("0");
  end g;

begin
  Put ("H");
  g;
  Put ("O");
end EH;

```